Improving multiple target tracking in structured environments using velocity priors

Rohan C. Loveland, Edward Rosten, and Reid Porter

Los Alamos National Laboratory, Los Alamos, NM, USA

ABSTRACT

In this paper, we present an algorithm for determining a velocity probability density prior from low frame rate aerial video of an urban area, and show how this may be used to aid in the multiple target tracking problem, as well as provide a foundation for the automated classification of urban transportation infrastructure. This is partially motivated by the difficulty of solving the multiple target tracking problem in instances with small targets and low sample rates. The utility of appearance cues is decreased in those circumstances where the targets are nearly identical, or appear to be so due to low sensor resolution relative to the overall system parameters. The performance of tracking methods based on motion models relying on target trajectory smoothness is similarly decreased when the sensor sampling rate is low relative to the target motion parameters. If the tracking is occurring in a structured environment, however, such as in the urban car tracking problem, some performance may be regained by incorporating prior knowledge of that environment into the tracking model. The velocity prior provides a means of describing this environmental knowledge.

Keywords: tracking, video, traffic, urban, transportation

1. INTRODUCTION

The increasingly widespread availability of high-resolution video sensors, coupled with improvements in accompanying storage media, processors, etc... have made the collection of aerial video for the purposes of tracking automobiles and monitoring traffic much more feasible in recent years.[1] Corresponding efforts for the acquisition of aerial video imagery of traffic data using helicopters for the flight platform have been conducted by Angel and Hickman,[2] and Hoogendoorn.[3] More recently, a data collect has been conducted in a joint effort between Los Alamos National Laboratory (LANL) and the University of Arizona, a frame of which (overlaid on a GoogleMaps image), is shown below in Fig. 1.

It is often desirable when collecting such video data to maximize the area covered. Unfortunately, limitations on storage/communications bandwidth and sensor resolution result in trade-offs between this and other system parameters. In those cases where maximum area coverage is a strong driver, the number of pixels on target, and the sampling rate, will both tend to be low. This in turn tends to reduce tracking performance, in that virtually all tracking algorithms are based on either: 1) *appearance cues*, which require enough pixels on target so that neighboring targets are distinguishable, or 2) *smooth trajectory assumptions*, which are broken if sampling rates fall too low with respect to the underlying spatial frequencies of the target trajectories.

In some environments, however, underlying rules apply to the target trajectories such that prior knowledge can be utilized to aid in the tracking process. In urban environments in particular, such rules are reflected by the existence of restricted regions (i.e. cars don't drive through walls), as well as high probabilities of certain speeds and directions of travel in allowed regions (i.e. traffic lanes have directions and speed limits).

In the following, we outline an algorithm that effectively allows the inference of these rules by building up a probability density function of the velocity at each pixel in the scene. This is done using track fragments, and takes advantage of the extremely high amount of information available to allow the application of a primitive association algorithm, from which only a relative few, high confidence track fragments are retained to build up the velocity probability densities. We then present the results and discuss how this provides advantages not only for future tracking algorithms, but also insight into the transportation structure of the urban environment.

Further author information: (Send correspondence to R.C.L.)
R.C.L.: E-mail: rohan@lanl.gov, Telephone: 1 505 667 4892

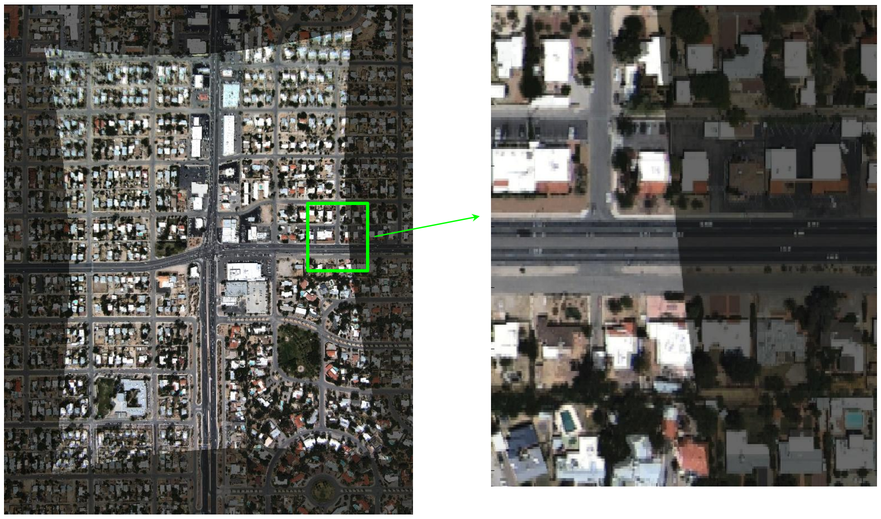
Figure 1. This shows a frame of imagery taken from the helicopter transparently superimposed over a region of the same image acquired from GoogleMaps. The region in the green box on the left is shown zoomed in on in the image on right. The irregular boundaries of the image are due to the radial distortion correction and registration processes.

2. DATA

The data used is the joint LANL/UofA data mentioned previously. A frame of this is shown above in Fig. 1. In this paper we use a 1024x1024 gray-scale central portion of the data to illustrate the algorithm. A brief description of some of the data parameters is given below.

Table 1. Data Parameters.

Original Image Size	2560x1920 pixels
Sampling Rate	$\sim$ 5 frames/second
Spatial Resolution	$\sim$ 0.75 ft./pixel
Helicopter Height (AGL)	$\sim$ 3000 ft.
Field of View	$\sim 70°$
Misregistration Jitter	up to 15 pixels/frame

The original image size is large enough to be computationally difficult, so for the results shown below a 1024x1024 central section was used, decimated by a factor of 2 to reduce size and improve car detection results. The misregistration jitter is particularly notable - this can cause a car to seemingly move its entire length in 1/6th of a second, corresponding to $\sim$ 60 mph jumps between frames. The jitter is 0 drift, however, and so skipping frames partially ameliorates this problem, as does using the central portion of the image where the jitter is less. For this reason the prior is calculated in frame pairs with 2 frames skipped in between. The combined effect reduces the inter-frame jumps to $\sim$ 20 mph. The overall effect of the jitter is to 1) introduce noise into each track fragment, and 2) blur the velocity probability density locations. The jitter is the result of sudden shifts in helicopter pose, combined with the the camera having a 'rolling shutter'. We can infer from the jitter magnitude that the helicopter pose variations ranged up to 5 milliradians between frames.

3. ALGORITHM

The goal of the algorithm is to build up a velocity probability density for every location (i.e. pixel) in the imagery. The velocity probability densities may be used as priors, in the Bayesian sense, for individual car tracking, so we'll refer to the aggregration of them over the entire image as the 'prior'. The prior at any given pixel location is represented with a polar histogram, with 5 speed bins along the radial axis, and 8 direction bins at the various angles. (see Fig.4 for an illustration)

The prior is built up, over all available pairs of frames, by:

1. **Car Detection** - *finding the locations of all objects that look like cars,*
2. **Car Association** - *finding track fragments in a pair of frames (so matching a car's location in the first image with its location in the second),*
3. **Track Filtering** - *filtering out all but the few track fragments for which the confidence of correct association is high,*
4. **Track Interpolation** - *for each remaining track fragment, determining the corresponding velocity and locations the car passed through,*
5. **Histogram Accumulation** - *incrementing the appropriate velocity bin in each histogram of those locations.*

Each of these steps is explained further in a subsection below.

3.1 CAR DETECTION

The car detection process is difficult given the available level of spatial resolution. Fortunately, motion filtering can be used to compensate for the initial inclusion of a large number of false alarms. The car detection algorithm used here is correspondingly non-specific, being the Difference of Gaussian (DoG) based multi-scale blob detector, as described and implemented by Lowe.[4]

In brief, the detector convolves the standard centered 2-D Gaussian:

$$G(x_1, x_2, k_i\sigma) = \frac{1}{2\pi(k_i\sigma)^2} exp^{-({x_1}^2+{x_2}^2)/(2(k_i\sigma)^2)} \tag{1}$$

over an image $I(x_1, x_2)$ at varying scales of $k_i\sigma$, producing,

$$L(x_1, x_2, k_i\sigma) \equiv G(x_1, x_2, k_i\sigma) * I(x_1, x_2) \tag{2}$$

The differences between these convolved images are then taken (hence the DoG nomenclature).

$$D(x_1, x_2, k_i, k_j, \sigma) \equiv L(x_1, x_2, k_i\sigma) - L(x_1, x_2, k_j\sigma) \tag{3}$$

This produces a 3-D structure in scale-space (i.e. in (x_1, x_2, k)). All local maxima and minima of this 3-D structure are then found by comparing each point with it's 26 neighbors (i.e. 9 above, 8 in the same k-plane, and 9 below), and each of these scale-space extrema is taken as the center of a blob with an associated scale of k.[*] Intuitively, this process can be viewed as convolving matched filter masks to the imagery at different scales, with the strongest responses being found where the underlying image points are most similar to the masks. The masks themselves are differences of 2-d Gaussians , which is to say circular with positive centers and negative surrounds (i.e. the 'mexican hat').

An argument can be made for a fairly rigid scale filter on the blob sizes, since most cars are quite similar in size. In practice, however, it is empirically found that many of the locations found on cars are smaller than the actual car size, so that only an upper limit on scale is used. A zoomed in version of the corresponding results of the scale-filtered blob detector are shown on the left side of Fig. 2. The high number of false alarms is apparent.

These are reduced through motion filtering. Points which are stationary provide no contribution to the histogram accumulation, since the initial assumption is that all points in the image are stationary. The car

[*]The algorithm contains some refinements beyond what is described here; see Lowe[4] for a complete description.

detection hits are accordingly filtered using dithered normalized cross-correlation, centered on the location in the first frame.

In more detail, this is accomplished by:

1. *getting a car-sized template of pixels from the neighborhood surrounding the blob location in the first frame,*
2. *getting a second same-sized template from the neighborhood surrounding that same location in the second frame,*
3. *calculating a normalized cross-correlation (NCC) score, as described in eqn. 5,*
4. *repeating this for a neighborhood of locations in the second frame surrounding the original location (i.e. 'dithering' the first template),*
5. *finding the the maximum NCC score from this dithered neighborhood (i.e. the best match),*
6. *filtering out those blobs with a sufficiently high matching score.*

Normalized cross-correlation is a standard means of template matching in computer vision.[5–7] Formally, a square template centered at $\vec{x}$, of side length s, taken from image I, can be defined as:

$$T_{\vec{x},s,I} = \{I(x_1 - k, x_2 - l) | \vec{x} \equiv [x_1, x_2]^T, k, l \in [-s/2, s/2]\}. \tag{4}$$

Argument s is uniformly set to a car-sized 15 pixels, and is omitted from here on. The templates can be normalized by subtracting their respective means μ_T and standard deviations σ_T, with the normalized cross-correlation then defined as:

$$NCC(T_1, T_2) = \frac{\sum_{k,l}(T_1(k,l) - \mu_{T_1})(T_2(k,l) - \mu_{T_2})}{\sigma_{T_1}\sigma_{T_2}} \tag{5}$$

where T_1 is understood to be $T_{\vec{x_a},I_1}$ and T_2 is understood to be $T_{\vec{x_b},I_2}$, where $\vec{x_a}$ is the (fixed) blob location in the first frame and $\vec{x_b}$ is allowed to vary in the second.

We now find

$$\vec{x_b}^* \equiv \arg\max_{\vec{x_b}} NCC(T_{\vec{x_a},I_1}, T_{\vec{x_b},I_2}) \tag{6}$$

where $\vec{x_b}$ ranges over a region centered on $\vec{x_a}$ that is determined by jitter magnitude - in this case set to a radius of 10 pixels. Finally, we reject all blob locations where $NCC(T_{\vec{x_a},I_1}, T_{\vec{x_b}^*,I_2}) > k$, where k is a manually selected matching threshold. This procedure effectively eliminates those blobs that have a good match in the second image in roughly the same location. This is repeated in reverse (i.e. switching the roles of frames 1 and 2) for additional robustness. Fig. 2 shows the results of this on the right; most of the remaining car detector hits are on the street, though in the full (unzoomed) image some obvious false alarms (e.g. on the roofs) remain.

3.2 CAR ASSOCIATION

After the stationary cars have been filtered out a list of car locations is left for frame 1 and frame 2. Given that the sequential frames don't have much rotation or scale change between them, we reuse the image templates described above in eqn. 4 for descriptors. Some examples of the templates are shown in Fig.3. The descriptor of a given car in frame 1 is compared to all of the cars in frame 2 that are within the possible movement limit, assumed here to be a maximum of 100 pixels/second, corresponding to ∼75 mph, with the best match chosen for the association. Rather than using the NCC score for comparison, the template descriptor is viewed as high-dimensional vectors. Following Lowe's[4] SIFT work, a matching function, $m(...)$, is then defined using the arccosine of the dot products of the descriptors as an approximation to the Euclidean distance between the vectors,

$$|\vec{T_1} - \vec{T_2}| \simeq m(\vec{T_1}, \vec{T_2}) \equiv \arccos(\vec{T_1} \bullet \vec{T_2}). \tag{7}$$

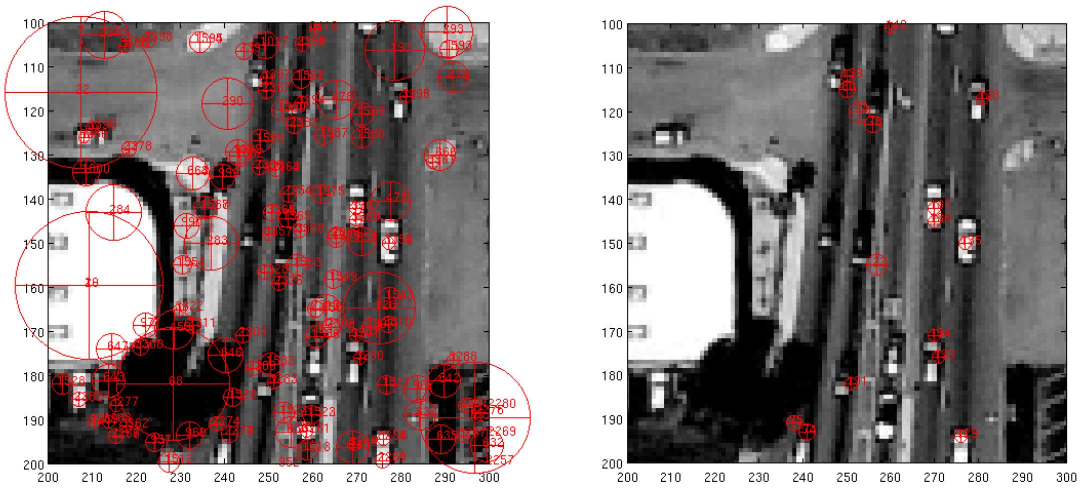

Figure 2. On the left of the figure a zoomed in part of the image is shown with the unfiltered hits from the blob detector marked with red circled crosshairs. The circle sizes are proportional to the scales of the corresponding blobs. The right side shows the hits filtered both by scale and motion.

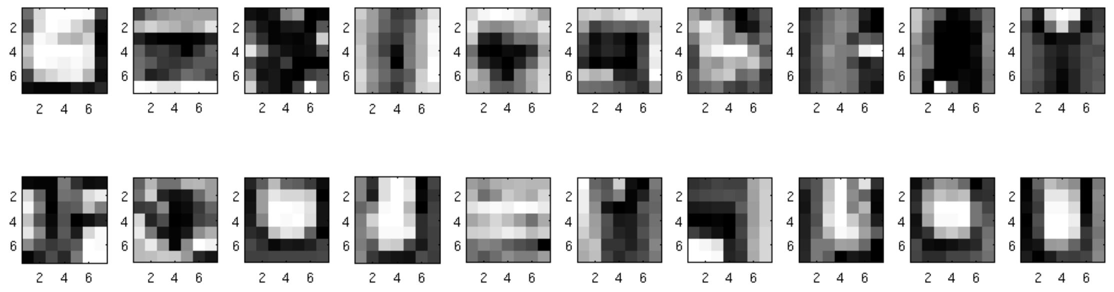

Figure 3. This shows examples of some of the car template descriptors (i.e. regions taken from the image neighborhoods centered on the car detector locations).

3.3 TRACK FILTERING

The possible matches are then sorted by score, and a match is only accepted if the ratio between the 1st and 2nd best matches is greater than a threshold of 0.6[†]. This ensures that the only matches accepted are those for which the confidence is high, due to the enforced low ambiguity. This consequently throws away a great number of correct matches. This is acceptable because the extremely large amount of information available will allow it. The ultimate result of this and the other filtering steps is to reduce some 3,500 original blob detector hits (on average) down to about 10 tracks retained per frame pair.

3.4 TRACK INTERPOLATION

Each track fragment specifies only the endpoints of the car trajectory, which is to say, where it started in frame 1 and ended up in Frame 2. We assume a constant velocity between these locations, so that each track fragment k has a corresponding speed s_k and direction d_k. We then calculate a set of points $\mathcal{N}_k$, defined to include those pixel locations passed through between the endpoints of track fragment k, expanded to further include approximately a half car width of points on both sides of the track fragment.

[†]this fairly sensitive parameter is left at the value set by Lowe

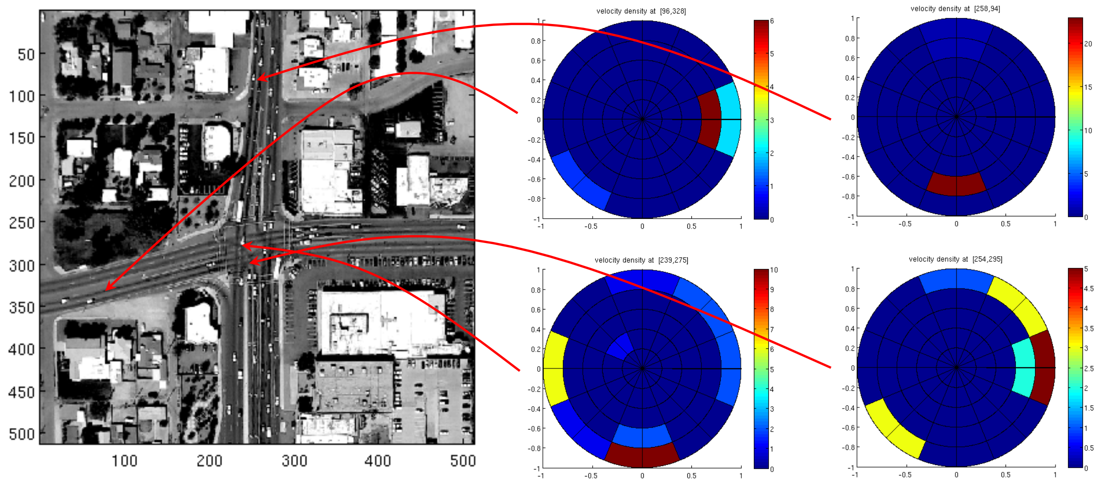

Figure 4. This shows the zoomed in center of the video data that the velocity prior was generated from on the left. On the right we have the polar histogram priors at 4 particular locations, with the red arrows showing the parts of the image that the histograms correspond to.

3.5 HISTOGRAM ACCUMULATION

A four dimensional data structure, $\mathbb{P}_{\tilde{\mathbf{x}},\mathbf{d},\mathbf{s}}$, is then allocated to accumulate the information from the track fragments. Let the total number of track fragments be K. Then we build up the histograms at each pixel by incrementing

$$\mathbb{P}_{\tilde{\mathbf{x}},\mathbf{d_k},\mathbf{s_k}} \Leftarrow (\mathbb{P}_{\tilde{\mathbf{x}},\mathbf{d_k},\mathbf{s_k}} + 1)\ \forall\, \vec{x} \in \mathcal{N}_k,\ k \in [1, K]. \tag{8}$$

This results in the building up of a 2-D (speed,direction) histogram at every point $\vec{x}$ that any track fragments pass through.

4. RESULTS

The results of building up the velocity prior can be viewed in several different ways. Fig. 4 shows the polar histograms at 4 different pixel locations, with red arrows indicating the particular pixel locations that each histogram corresponds to. The directions of the polar plots are oriented the same as the plot on the left, while the radii of the bins indicate speed, with the outer ring being the fastest. The colors show the number of counts in each of the bins, with the scale being shown by the accompanying color bars on the right of the histograms. Red is highest and dark blue is lowest.

Knowing that traffic rules include driving on the right for Tucson, the histograms indicate what one would expect for each of the locations in question, though the inclusion of some counts due to incorrect track fragments can also be seen. The upper left and upper right histograms show east-bound and south-bound streets, and have dark red peaks clearly indicating this. The lower left histogram with the yellow west-bound peak and the red south-bound peak also shows what one would expect to see in that particular part of the intersection. The lower right histogram is clearly the most complex, with east-bound traffic going straight, as well as south-bound left and u-turners.

An alternative way of viewing the data is to find the mode of each histogram at each location, resulting in numbers representing the most common speed and direction at each location. The results of this are shown in Fig. 5 and Fig. 6. The speeds look correct, and show that drivers tend to leave intersections faster than they approach them. The directions also look correct, in that the opposite sides of the road have opposite colors. This view also reveals the large number of incorrect track fragments that passed the filters.

These results were generated using 1126 frames, corresponding to about about 3 1/2 minutes of video. Obviously, a more complete velocity prior can be built up using more data.

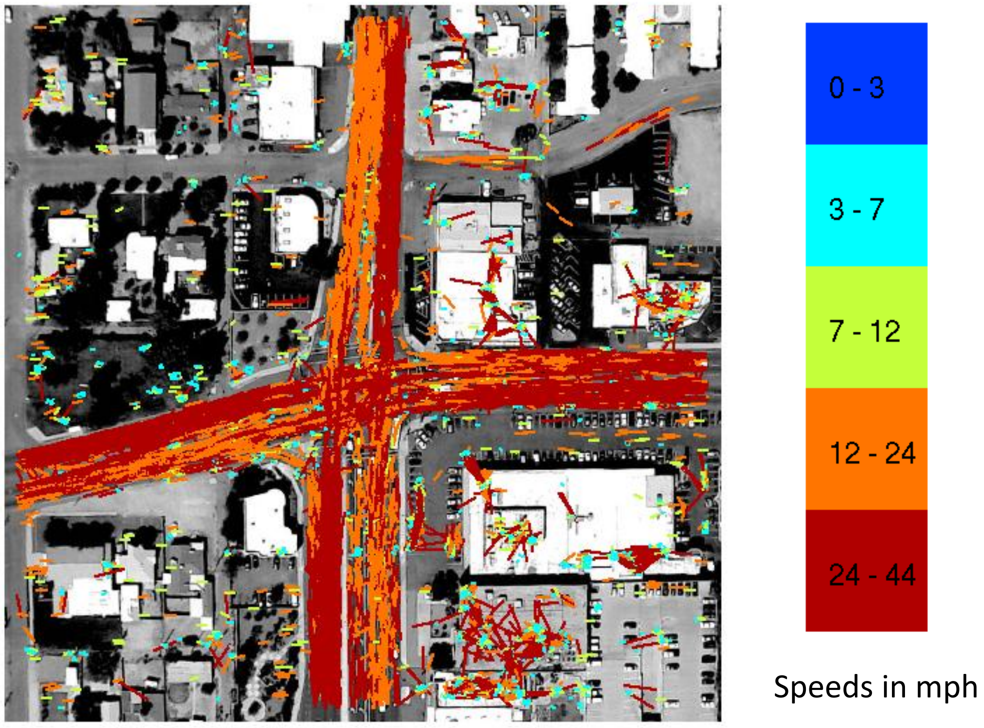

Figure 5. This shows the results of taking the mode of the velocity prior at each location and plotting the speed. The bar on the right shows the speeds in mph that the colors correspond to.

5. CONCLUSION

The results are encouraging in that they match what we would expect based on our knowledge of traffic regulations and structure. It is clear that false matches and jitter both degrade the velocity prior; we hope to address both of these issues in future work, as well as to use more of the available data both spatially and temporally. The prior is expected to be useful in tracking applications, where the prior may be used to restrict searches to drivable areas and to assign higher probability to matches that are in accordance with lane directions. The prior may also be used to gain information about urban transportation infrastructure and traffic flow.

ACKNOWLEDGMENTS

The cooperation and encouragement of Drs. Mark Hickman and Pitu Mirchandani of the University of Arizona is very gratefully acknowledged. Thanks also to Dr. David Lowe of the University of British Columbia for making his SIFT code available online. Partial funding for the data collection and reduction for this effort has been supplied by Los Alamos National Laboratory, the National Nuclear Security Administration, and the U.S. Department of Energy.

REFERENCES

1. R. Kumar, H. Sawhney, S. Samarasekera, S. Hsu, H. Tao, Y. Guo, K. Hanna, A. Pope, R. Wildes, D. Hirvonen, M. Hansen, and P. Burt, "Aerial video surveillance and exploitation,," *Proceedings of the IEEE* **89 no. 10**, pp. 1518–1539.
2. A. Angel, M. Hickman, P. Mirchandani, and D. Chandnani, "Methods of traffic data collection, using aerial video,," *Proceedings. The IEEE 5th International Conference on Intelligent Transportation Systems* **89 no. 10**, pp. 31–36, 2002.

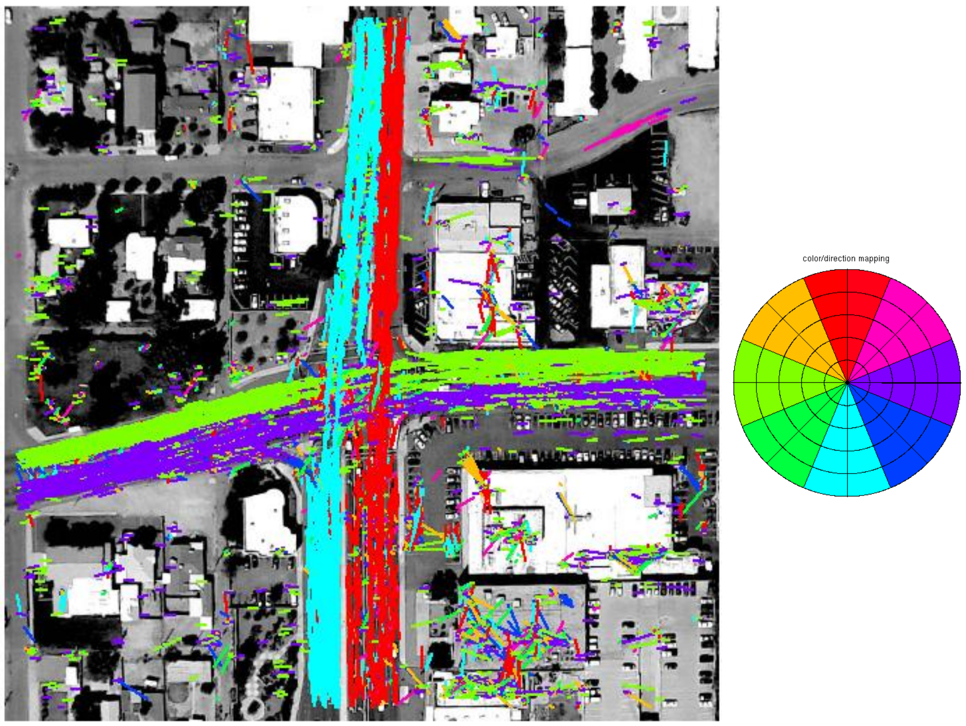

Figure 6. This shows the results of taking the mode of the velocity prior at each location and plotting the direction. The pie on the right shows the directions that the colors correspond to.

3. S. Hoogendorn, H. Van Zuylen, M. Schreuder, B. Gorte, and G. Vosselman, "Microscopic traffic data collection by remote sensing,," *Transportation Research Board Annual Meeting CD-ROM* , 2003.
4. D. Lowe, "Distinctive image features from scale-invariant keypoints," in *International Journal of Computer Vision*, **20**, pp. 91–110, 2003.
5. D. Ballard and C. Brown, *Computer Vision*, Prentice-Hall, New Jersey, 1982.
6. R. Gonzalez and R. Woods, *Digital Image Processing (3rd Edition)*, Addison-Wesley, Reading, Mass., 1992.
7. R.Duda and P.Hart, *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.